uTensor: A Graph to C++ Code Generator

Neil Tan, Kazami Hsieh, Dboy Liao, Michael Bartling

uTensor is a user-friendly workflow that enables machine learning inferencing on microcontrollers (MCUs) built on top of TensorFlow and Mbed. The ability to perform ML inferencing on MCUs enables cost-effective, lower-power and smarter edge devices. Other benefits include reducing latency, conserving bandwidth and improving privacy. However, until recently, deploying ML models on the edge has been challenging due to the lack of tools and MCUs' resource constraints. uTensor is one of the first frameworks that bring machine learning inferencing onto the edge. We tackled this with a graph-to-source-code generation approach.

uTensor consist of a standalone C++ runtime library and a code generator. The C++ library contains common kernel functions required in neural network evaluations. The code generator automates the process of creating C++ inference functions from models.

During the code generation, a model is parsed into uTensor intermediate graph representation (IR). Multiple graph-rewriting passes are then applied to the IR. These passes including weight quantization, dropout removal, node insertion, replacement, and fusion. The end result is an inference-optimized IR readily be used for C++ code generation. The output of the code generator, C++ source files, can be easily copy-and-paste into existing embedded projects. From a developer's perspective, it only takes one function call to invoke the inferencing on the MCU.

With this approach, we have trained MLP and CNN models on MNIST and CIFAR10 datasets in TensorFlow. The uTensor generated source files are imported into the Mbed projects, compiled and deployed on various MCU targets. An interactive touchscreen handwritten-digit recognizer has been successfully created with this workflow on a Cortex-M4 target. When tinyML takes place, we expect to see these models running on even smaller targets such as Cortex-M0 alike.

uTensor is also a powerful graph manipulation tool. It is written in Python prioritizing extensibility and customizability. We hope to empower data scientists and machine-learning practitioners to rapidly verify their latest ideas and advance the progress in edge computing.